(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 449 661 B1**

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication of patent specification : **08.11.95 Bulletin 95/45**

(51) Int. Cl.[6] : **G06F 9/38**

(21) Application number : **91302832.0**

(22) Date of filing : **28.03.91**

(54) Computer for Simultaneously executing plural instructions.

(30) Priority : **30.03.90 JP 84607/90**

(43) Date of publication of application : **02.10.91 Bulletin 91/40**

(45) Publication of the grant of the patent : **08.11.95 Bulletin 95/45**

(84) Designated Contracting States : **DE FR GB IT**

(56) References cited :
COMPCON 90: THIRTY-FIFTH IEEE COMPUTER SOCIETY INTERNATIONAL CONFERENCE 2 March 1990, LOS ALAMITOS, CA, US pages 232 - 240; S. MCGEADY: 'The i960CA SuperScalar Implementation of the 80960 Architecture'
COMPCON 89: THIRTY-FOURTH IEEE COMPUTER SOCIETY INTERNATIONAL CONFERENCE 3 March 1989, pages 13 - 17; G. HINTON: '80960 - Next Generation'
IEEE TRANSACTIONS ON COMPUTERS. vol. C-32, no. 5, May 1983, NEW YORK US pages 425 - 438; J. E. REQUA ET AL.: 'The Piecewise Data Flow Architecture: Architectural Concepts'

(56) References cited :
*** JP-A-61245239 * IEEE Transactions on Computers, vol. C-27, no.3, March 1978, US, pages 270-275; HIGHBIE: "Overlapped Operation with Microprogramming" * Proceedings of the International Conference on Parrallel Processing, vol. 1, August 1988, US, pages 426-429; WANG, WU: "Distributed Instruction Set Computer"**

(73) Proprietor : **KABUSHIKI KAISHA TOSHIBA**
72, Horikawa-cho,
Saiwai-ku
Kawasaki-shi, Kanagawa-ken 210, Tokyo (JP)

(72) Inventor : **Saito, Mitsuo, c/o Intellectual Property Div.**
Toshiba Corporation,
1-1 Shibaura
1-chome, Minato-ku, Tokyo (JP)
Inventor : **Minagawa, Kenji, c/o Intellectual Property Div.**
Toshiba Corporation,
1-1 Shibaura
1-chome, Minato-ku, Tokyo (JP)
Inventor : **Aikawa, Takeshi, c/o Intellectual Property Div.**
Toshiba Corporation,
1-1 Shibaura
1-chome, Minato-ku, Tokyo (JP)

(74) Representative : **Shindler, Nigel**
BATCHELLOR, KIRK & CO.
2 Pear Tree Court
Farringdon Road
London EC1R 0DS (GB)

EP 0 449 661 B1

Note : Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid (Art. 99(1) European patent convention).

Description

The present invention relates to a computer having a plurality of types of operation units, where the plurality of operation units can execute instructions simultaneously.

In conventional computer systems, the computer reads out instructions one by one from a main memory and executes the instructions in order. Even though a conventional computer has a plurality of types of operation units, only one operation unit executes one instruction during any one clock cycle. Thus, the processing speed for conventional computers is low.

Recently, the plurality of types of operation units in the computer have been put to practical use. The computer reads out plural instructions from the main memory and executes the plurality of instructions simultaneously. A representative method is called "Super Scalar" see for example COMPCON 90: Thirty-Fifth IEEE Computer Society International Conference, 2nd March 1990, Los Alamitos, CA, U.S. pages 232-240; S. McGready: "The 1960CA SuperScalar Implementation of the 80960 Architecture". In this method, the computer, the computer includes a cache memory. First, a subset of instructions stored in main memory are read out from the main memory to the cache memory. The number of the instructions to be executed simultaneously is fixed in advance. Then, the subset of the Instructions are read out from the cache memory. The instructions are examined to determine the type of the operation unit required by each instruction, and the possibility of simultaneous execution. Ones of the subset of instructions are respectively supplied to corresponding operation units according to the decision result, and the operation units executes the corresponding instructions. The decision process is performed for several subsets of instructions read out from the cache memory in order.

However, in the "Super Scalar" method, a decision is made for the subsets of instructions when the instructions are read out from the cache memory to the operation unit. The cache memory stores instructions that are used repeatedly. Therefore, the system repeatedly performs the deciding step whenever the instructions are read out from the cache memory. In short, the cycle time for executing the instruction increases because of the repeated decision process.

Another system for improving the speed of operation is suggested in Proceedings of the International Conference on Parallel Processing, vol. 1, August 1988, U.S. pages 426-429, Wang, WU: "Distributed Instruction Set Computer" which describes a multi-bit flag attached to each instruction indicating the number of cycles the instruction has to wait for the data to become ready in other execution units. The instructions are stored together with their flags in a cache memory.

Accordingly, it is one object of the present invention to provide a computer for simultaneously executing plural instructions, where the cycle time for executing the instruction of the computer does not increase.

It is another object of the present invention to provide a computer for simultaneously executing plural instructions, where the computer does not decide the possibility of simultaneous execution for the plural instructions when the instructions are read out from the cache memory to the operation unit.

These and other objects of the present invention are accomplished by deciding whether simultaneous execution is possible for the plural instructions when they are read out from the main memory to the cache memory. The plural instructions and decision result are stored in the cache memory. The decision process is performed for several subsets of the plural instructions read out from the main memory in order. Then, the plural instructions are immediately assigned to corresponding operation units according to the decision result. As a result of this arrangement, the repeated decision process for plural instructions is not necessary when the plural instructions are read out from the cache memory to the operation unit.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a block diagram of a computer for simultaneously executing plural instructions according to the present invention;

Figure 2 shows a format of the plural instructions and the decision result according to the present invention;

Figure 3 shows a construction of a instruction contention analysis section according to the present invention;

Figure 4 shows a construction of a distribution matrix according to the present invention;

Figure 5 shows a construction of a score board stored in a memory according to the present invention; and

Figure 6 shows an example of a score board for simultaneously executing the plural instructions according to the present invention.

DESCRIPTION OF PREFERRED EMBODIMENTS

Figure 1 shows a block diagram of a computer for simultaneously executing plural instructions. A main memory 1 stores a large number of the instructions in predetermined order. A subset of the instructions are read out from the main memory 1 to a cache memory 3 inside a processor 2 and the instructions are respectively supplied to an operation unit 4-9 where they are executed. Figure 1 shows an embodiment of the present invention including a floating-adder (F-ADD) 4, a floating-multiplier (F-MUL) 5, two integer-

arithmetic logical units (I-ALU) 6 and 7, a branch-control unit 8 and an exception process unit (EXCEPTION) 9. A distribution matrix 10 supplies the plural instructions to the operation units 4-9 according to the type of operation performed by the instructions.

The subset of plural instructions are read out from the main memory 1 to the cache memory 3 at the beginning.

For example, several subsets of four instructions, which are to be repeatedly used, are read out from the main memory 1 in order.

The possibility of simultaneous execution for the four instructions is decided when they are read out from the main memory 1 to the cache memory 3 according to the present invention. The invention includes a decision means, which is embodied as an instruction contention analysis section 11. Instruction contention analysis section 11 determines the type of operation for the four instructions by referring to their operation codes and decides corresponding operation units for the four instructions respectively. Then the section 11 determines whether the types of operations are mutually different or not. If at least two of the types of operations are the same, the section 11 adds a flag to the corresponding instruction. The flag indicates that the corresponding instruction is not to be executed during the same clock cycle. The instruction contention analysis section 11 examines whether of the same operation units corresponds to at least two of four instructions or not. If, for example, two instructions requiring the same operation unit are detected, the section 11 sets a flag of the second instruction that corresponds to the first instruction to "1." The flag belongs to the lower priority instruction of the second instruction and requires that the second instruction wait to be executed by the operation unit. The priority of instruction is defined by an address of the four instructions. The larger the address of instruction, the lower the priority of the instruction. Accordingly, the instruction corresponding to the first priority among the four instructions always has a flag value of "0". (In some embodiments, the flag bit corresponding to the first priority is omitted). In short, three instructions corresponding to the second, third, fourth priority include a flag bit for indicating the contention of the operation unit.

Figure 2 shows a format of a decision result output from the instruction contention analysis section 11. The format includes three bits for each instruction to indicate the type of operation for the instruction and one bit for each instruction to indicate the contention of the operation unit for one instruction (except for the first priority instruction). As shown in Figure 2, a preferred embodiment includes seven types of operations. Therefore, three bits are necessary to indicate the type of operation. If an instruction must wait for executing because of overlap of the operation unit, the bit indicating the contention of the operation unit is set to "1". The format has fifteen bits in total, twelve bits of which respectively indicate the type of operation for four instructions (#0, #1, #2, #3) and three bits of which (MARK) respectively indicate the contention of the operation unit for three instructions (#1, #2, #3). There is no flag shown for instruction #0. Four instructions and the decision result (fifteen bits in total) are stored in the cache memory 3. The above-mentioned decision process is performed for several subsets of four instructions in the main memory 1 whenever they are read out from the main memory to the cache memory in order.

Then, four instructions and the decision result are read out from the cache memory 3 to an instruction buffer 12. The distribution matrix 10 (assignment means) reads out the four instructions with the decision result from the instruction buffer 12. The distribution matrix 10 supplies the four instructions to a corresponding operation unit 4-9 according to the decision result. At this moment, the distribution matrix 10 refers to a program counter (not shown in the Figure). The program counter indicates the address of a previously executed final instruction. The distribution matrix 10 confirms the end of execution of all four previous instructions, referring to the program counter. In applicant's invention, it is not necessary for four instructions a type of operation and contention to be decided when the instructions are read out from the cache memory to the operation unit. The distribution matrix 10 is able to immediately supply the four instructions to the operation unit according to the decision result.

After the four instructions are supplied, the operation units 4-9 read out the date from the register that is necessary for executing the instruction. (The register section 13 consists of nine registers in this example.) (See Fig. 6) At this time, the operation unit analyzes the register number to which the result of the operation is to be written according to the instruction and sets a flag corresponding to the register number in the score board 14 to "1" according to the type of operation (see Fig. 6). The score board 14 stores information that indicates whether or not the register is monopolized by the operation unit for several cycles. If the value of a register for executing the instruction is detected for the operation unit to be useless (monopolized by the other operation unit) according to the score board 14, the operation unit abandons the executed result and executes the same instruction again for next cycle.

Figure 3 shows a construction of the instruction contention analysis section 11 (decision means). As shown in Figure 3, the instruction contention analysis section 11 includes the read-out buffer 31, the four decoders 32, 33, 34, 35 and a decision result generation section 36. The read-out buffer 31 temporarily stores four instructions read-out from the main memory 1. The four instructions are respectively transfer-

red from the read-out buffer 31 to the cache memory 3 and the four decoders 32, 33, 34, 35. The first (highest) priority instruction is transferred to the decoder 32. The fourth (lowest) priority instruction is transferred to the decoder 35. The decoders 32, 33, 34, 35 analyze the type of operation unit required to execute the instruction and the decoders 33, 34, 35 examine whether the same operation unit is required by the higher priority instruction or not. The three decoders 33, 34, 35 respectively receive the decode result of higher priority decoders 32, 33, 34 in order and decide whether the received decode result indicates contention of the operation unit or not. If the type of operation unit of the decoder is the same as the type of the operation unit of the higher priority instruction, the decoder sends a mark bit of "1". The decision result generation section 36 receives the type of operation unit and the mark bit indicating whether there is contention for the operation unit (mark bit) from the four decoders 32, 33, 34, 35 or not. The decision result generating section 36 generates the decision result as shown in Figure 2. Then the decision result corresponding to the four instructions is stored in the cache memory 3.

Figure 4 shows construction of the distribution matrix 10 (assignment means). The matrix 10 includes four decoders 41, 42, 43, 44 and a switch matrix 45. Each decoder receives a type of operation unit and a contention mark from the instruction buffer 12. The decoder 41 (highest priority) receives only the type of operation unit. If the contention mark is "0", the decoder opens a gate of a switch matrix 45 corresponding to the type of operation unit. (The decoder 41 opens the gate unconditionally.) If the contention mark is "1", the decoder does not open the gate corresponding to the type of operation unit and opens the gate at a next cycle. In short, the gates of switch matrix 45 are opened by the execution timing according to the type of operation unit and the contention mark. Therefore, the plural instructions are effectively supplied to a corresponding operation unit without excess overhead time.

Figure 5 shows a construction of the score board 14. The score board 14 includes a score board register 51 and four judgement circuits 52, 53, 54, 55, which are cyclically connected by corresponding bits. (A preferred embodiment has nine registers in register section 13.) The score board register 51 consists of bits corresponding to the registers in the register section 13, respectively. At the beginning, the score board 51 stores information concerning whether the previous four instructions had completed executing or not. If all bits of the score board register 51 are "0", the previous four instructions completed executing. If at least one bit of the score board register 51 is "1", the previous instruction was not finished executing. When all bits of the score board register 51 are "0", the four judgement circuits begin processing. The judgement circuit 52 receives the first priority instruction through the operation unit and detects the register number for writing the operation result, where the operation requires plural clock cycles. The judgement circuit 52 writes "1" in the bit corresponding to the register number if the operation requires plural clock cycles and sends the completion signal to the corresponding operation unit. The judgement circuit 53 receives the second priority instruction through the operation unit and detects the register number for writing the operation result, where the operation requires plural clock cycles and register number for reading data which is necessary for execution. The judgement circuit 53 receives the bit of the score board register 51 including the judgement result of the judgement circuit 52. If the bit corresponding to the register number for reading is set to "1", the judgement circuit 53 sends the wait signal to the corresponding operation unit. If the bit corresponding to the register number for reading is reset to "0", the judgement circuit 53 sends the completion signal to the corresponding operation unit. The judgement circuit 53 writes "1" in the bit corresponding to the register number for writing if the operation requires plural clock cycles. The judgement circuit 54 (corresponding to the third priority instruction) and the judgement circuit 55 (corresponding to the fourth priority instruction) perform the same processing as the judgement circuit 53. (The bit "1" of the score board register 51 is reset to "0" when the operation unit writes the operation result in the corresponding register.)

Figure 6 shows an example of the score board register 14 over time for simultaneously executing plural instructions. This example assumes that two instructions are simultaneously executed. As shown in an upper portion of Figure 6, three groups of two instructions (I1, I2) (I3, I4) (I5, I6) are executed in order.

When first two instructions (I1, I2) are executed at clock cycle 1, all bits of the score board register are reset to "0". The read register (r1, r5) of the instruction (I2) is not different from the write register (r3) of the instruction (I1). Therefore, two instructions (I1, I2) are able to be executed simultaneously at clock cycle 1. However, the instruction (I1) is an add operation, which requires one clock cycle, and the instruction (I2) is a multiple operation, which requires three clock cycles. Therefore, the bit corresponding to the register (r4), which is a write register of the instruction (I2), is set to "1". When the next two instructions (I3, I4) are executed at clock cycle 2, the first instruction (I3) is immediately executed because the read register (r3, r5) is reset to "0". But the bit corresponding to the register (V4), which is the read register of the instruction (I4), is already set to "1". Therefore, the instruction (I4) waits to be executed. In short, when the execution of the instruction (I2) is finished at clock cycle 3, the bit "1" corresponding to the register (r4) is reset to "0".

After the bit is reset, the instruction (I4) is immediately executed at clock cycle 4. At this time, the instruction (I4) is a multiple operation which requires three clock cycles. Therefore the bit corresponding to the register (r7), which is the write register of the instruction (I4), is set to "1".

When last two instructions (I5, I6) are executed at clock cycle 5, the bit corresponding to the register (r7), which is a read register of the first instruction (I5), was already set to "1". Therefore, the instructions (I5, I6) wait until the execution of the instruction (I4) is finished. After the bit "1" corresponding to the register (r7) is reset at clock cycle 6, the two instructions (I5, I6) are executed simultaneously at clock cycle 7.

The described embodiment contains a subset of four instructions. Other embodiments may contain a different number of instructions in a subset.

Claims

1. A computer for simultaneously executing plural instructions, comprising:

   plural types of operation units (4, 5, 6, 7, 8, 9) each for executing one of a plurality of corresponding types of instructions, and main memory (1) for pre-storing a plurality of instructions,

   decision means (11) for deciding the types of operation unit for each instruction of a subset of the plural instructions according to the types of the subset of the plural instructions and for deciding a possibility of simultaneous execution of the subset of the plural instructions wherein the decision of the decision means is called a decision result;

   cache memory (3) for temporarily storing the subset of the plural instructions and the decision result; and

   assignment means (10) for respectively assigning ones of the subset of the plural instructions to corresponding types of the operation units according to the decision result stored in the cache memory (3),

   characterised in that

   the decision means (11) decide on said subset of the plural instructions when the subset is read from the main memory (1), and send said decision result to said cache.

2. The computer for simultaneously executing plural instructions according to claim 1, wherein the decision means (11) determines that the plural instructions are to be simultaneously executed when the type of operations are mutually different.

3. The computer for simultaneously executing plural instructions according to claim 1, wherein the decision means (11) determines that the plural instructions are not to be simultaneously executed when at least two of the types of operations are the same.

4. The computer for simultaneously executing plural instructions according to claim 3, wherein the decision means determines that the plural instructions are to be simultaneously executed in case that number of the plural instructions whose types of operations are the same is less than number of the operation units whose type corresponds to the type of the plural instructions.

5. The computer for simultaneously executing plural instructions according to claim 1, wherein the assignment means (10) is a distribution matrix for supplying an instruction to a corresponding operation unit according to the type of the operation unit and the possibility of simultaneous execution.

6. The computer for simultaneously executing plural instructions according to claim 1, wherein the plural instructions are repeatedly assigned to the corresponding operation units (4, 5, 6, 7, 8) by the assignment means (10) and executed by the corresponding operation units.

7. The computer for simultaneously executing plural instructions according to claim 1, further comprising a plurality of registers (13), where information in the plurality of registers is accessed by the operation units during execution.

8. The computer for simultaneously executing plural instructions according to claim 7, wherein execution of a multiple clock cycle instruction by an operation unit causes at least one register to be monopolized for more than one clock cycle, further comprising a score board (14), which has flags corresponding to respective ones of the registers for indicating whether the registers are monopolised by the execution of an instruction or not.

9. The computer for simultaneously executing plural instructions according to claim 8, wherein one of the flags in the score board (14) is set, corresponding to a monopolised register, according to a priority of the plural instructions.

10. The computer for simultaneously executing plural instructions according to claim 9, wherein an executed result is produced by a multiple clock cycle instruction and wherein the operation unit decides to write the executed result in the monopolised register and to reset the flag corresponding to the monopolised register in the score board

(14).

11. A computer for simultaneously executing plural instructions according to claim 1, wherein several subsets of the plural instructions are read out from the main memory (1) in order.

Patentansprüche

1. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen, mit:
mehreren Arten von Operationseinheiten (4, 5, 6, 7, 8, 9), von denen jede eine einer Vielzahl zugehöriger Befehlsarten ausführt, und mit einem Hauptspeicher (1) zur Vorspeicherung einer Vielzahl von Befehlen;
Entscheidungseinrichtungen (11), um für jeden Befehl einer Untergruppe der Vielzahl von Befehlen entsprechend der Art der Untergruppen der Vielzahl von Befehlen die Arten von Operationseinheiten zu bestimmen und um eine Möglichkeit gleichzeitiger Ausführung der Untergruppe der Vielzahl von Befehlen zu bestimmen, wobei die Entscheidung der Entscheidungseinrichtungen als ein Entscheidungsergebnis bezeichnet wird;
einem Zusatzspeicher (3), um die Untergruppe der Vielzahl von Befehlen und das Entscheidungsergebnis zeitweise zu speichern; und
Zuordnungseinrichtungen (10), um entsprechend dem im Zusatzspeicher (3) gespeicherten Entscheidungsergebnis jeweils einer der Untergruppen der Vielzahl von Befehlen zugehörige Arten der Operationseinheiten zuzuordnen;
dadurch gekennzeichnet, daß
die Entscheidungseinrichtungen (11) bezüglich der Untergruppe der Vielzahl von Befehlen entscheiden, wann die Untergruppe aus dem Hauptspeicher (1) ausgelesen und das Entscheidungsergebnis zum Zwischenspeicher geleitet wird.

2. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 1, bei dem die Entscheidungseinrichtung (11) bestimmt, daß die Vielzahl von Befehlen gleichzeitig auszuführen ist, wenn die Arten von Operationen voneinander verschieden sind.

3. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 1, bei dem die Entscheidungseinrichtung (11) bestimmt, daß die Vielzahl von Befehlen nicht gleichzeitig auszuführen ist, wenn zumindest zwei der Arten von Operationen gleich sind.

4. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 3, bei dem die Entscheidungseinrichtung bestimmt, daß die Vielzahl von Befehlen für den Fall gleichzeitig auszuführen ist, daß die Zahl der Vielzahl von Befehlen, deren Arten von Operationen gleich sind, kleiner ist als die Zahl von Operationseinheiten, deren Art der Art der Anzahl von Befehlen entspricht.

5. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 1, wobei die Zuordnungseinrichtung (10) eine Zuordnungsmatrix ist, um einer zugehörigen Operationseinheit gemäß der Art der Operationseinheit und der Möglichkeit gleichzeitiger Ausführung einen Befehl zuzuleiten.

6. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 1, bei dem die Vielzahl von Befehlen durch die Zuordnungseinrichtungen (10) wiederholt den zugehörigen Operationseinheiten (4, 5, 6, 7, 8) zugeordnet und durch die zugehörigen Operationseinheiten ausgeführt werden.

7. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 1, der weiterhin eine Vielzahl von Registern (13) enthält, wobei während der Ausführung durch die Operationseinheiten auf Informationen der Vielzahl von Registern zugegriffen wird.

8. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 7, bei dem durch die Ausführung eines Befehls mit mehreren Taktschritten durch eine Operationseinheit bewirkt wird, daß zumindest ein Register für mehr als einen Taktschritt belegt ist, und außerdem eine Auswertetafel (14) aufweist, die entsprechenden Registern zugehörige Flaggen enthält, um anzuzeigen, ob die Register durch die Ausführung eines Befehls belegt oder nicht belegt sind.

9. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 8, wobei entsprechend einer Priorität der Vielzahl von Befehlen eine der Flaggen in der Auswertetafel (14) gemäß eines belegten Registers gesetzt ist.

10. Rechner zum gleichzeitigen Ausführen einer Vielzahl von Befehlen nach Anspruch 9, bei dem durch einen Befehl mit mehreren Taktschritten ein ausgeführtes Ergebnis erzeugt wird und wobei die Operationseinheit entscheidet, ob das ausgeführte Ergebnis in das belegte Register geschrieben und die dem belegten Register entsprechende Flagge in der Auswertetafel (14) zurückgesetzt werden soll.

11. Rechner zum gleichzeitigen Ausführen einer Viel-

zahl von Befehlen nach Anspruch 1, bei dem mehrere Untergruppen der Vielzahl von Befehlen nacheinander aus dem Hauptspeicher (1) ausgelesen werden.

Revendications

1. ordinateur pouvant exécuter plusieurs instructions simultanément comprenant:-

   plusieurs types d'unités d'opérations (4, 5, 6, 7, 8, 9), chacune de ces unités d'opération exécutant une instruction parmi plusieurs instructions, et une mémoire principale (1) pour stocker préalablement une pluralité d'instructions,

   un moyen de décision (11) pour décider le type d'unité d'opération correspondant à chaque instruction du sous-groupe d'instructions en fonction du type de sous-groupe d'instructions et pour décider une possibilité d'exécution simultanée du sous-groupe de plusieurs instructions dans lequel ordinateur la décision générée par le moyen de décision s'appelle un résultat de décision;

   une mémoire cache (3) pour stocker temporairement le sous-groupe de plusieurs instructions et le résultat de décision; et

   un moyen d'affectation (10) pour affecter respectivement chacune des instruction du sous-groupe de plusieurs instructions aux types correspondants d'unités d'opération en fonction du résultat de décision stocké dans la mémoire cache (3),

   caractérisé par le fait que le moyen de décision (11) prend une décision correspondant au sous-groupe de plusieurs instructions lorsque le sous-groupe d'instructions est lu en mémoire principale (1) et envoie le résultat de décision dans la mémoire-cache.

2. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 1, dans lequel le moyen de décision (11) détermine que ces instructions multiples doivent être exécutées simultanément lorsque les types d'opérations sont différent les uns par rapport aux autres.

3. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 1, dans lequel le moyen de décision (11) détermine que les instructions multiples ne doivent pas être exécutées simultanément lorsqu'au moins deux des types d'opération sont les mêmes.

4. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 3, dans lequel le moyen de décision détermine si oui ou non les instructions doivent être exécutées simultanément lorsque le nombre d'instructions multiples dont les types d'opérations sont les mêmes est inférieur au nombre d'unités d'opération dont le type correspond au type des instructions multiples.

5. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 1, dans lequel le moyen d'affectation (10) est une matrice de distribution qui envoie d'une part une instruction à une unité d'opération correspondante et d'autre part indique la possibilité d'exécution simultanée.

6. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 1, dans lequel les instructions multiples sont affectées de manière répétitive aux unités d'opération correspondantes (4, 5, 6, 7, 8) par le moyen d'affectation (10) et exécutées par les unités d'opération correspondantes.

7. Ordinateur pouvant exécuter simultanément des instructions multiples selon la revendication 1, comprenant en outre une pluralité de registres (13) dont les informations qu'ils contiennent sont utilisées par les unités d'opérations pendant l'exécution des instructions.

8. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 7, dans lequel l'exécution d'une instruction de multiplication prend plus d'un cycle d'horloge et pendant cette durée au moins plus d'un registre est monopolisé, comprenant en outre une matrice de priorité (14) qui a des drapeaux correspondant aux registres respectifs pour indiquer si oui ou non ces registres sont monopolisés par l'exécution d'une instruction.

9. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 8, dans lequel un des drapeaux de la matrice de priorités (14) correspondant au registre monopolisé est mis à « 1 », en fonction du niveau de priorité accordé aux instructions multiples.

10. Ordinateur pouvant exécuter simultanément plusieurs instructions selon la revendication 9, dans lequel un résultat exécuté a été produit par une instruction qui prend plusieurs cycles d'horloge et dans lequel l'unité d'opération décide d'écrire le résultat exécuté dans le registre monopolisé et de réinitialiser le drapeau correspondant au registre monopolisé de la matrice de priorités (14).

11. Ordinateur pouvant exécuter de manière simultanée plusieurs instructions selon la revendication

1, dans lequel plusieurs sous-groupes d'instructions multiples sont lus dans l'ordre dans la mémoire principale (1).

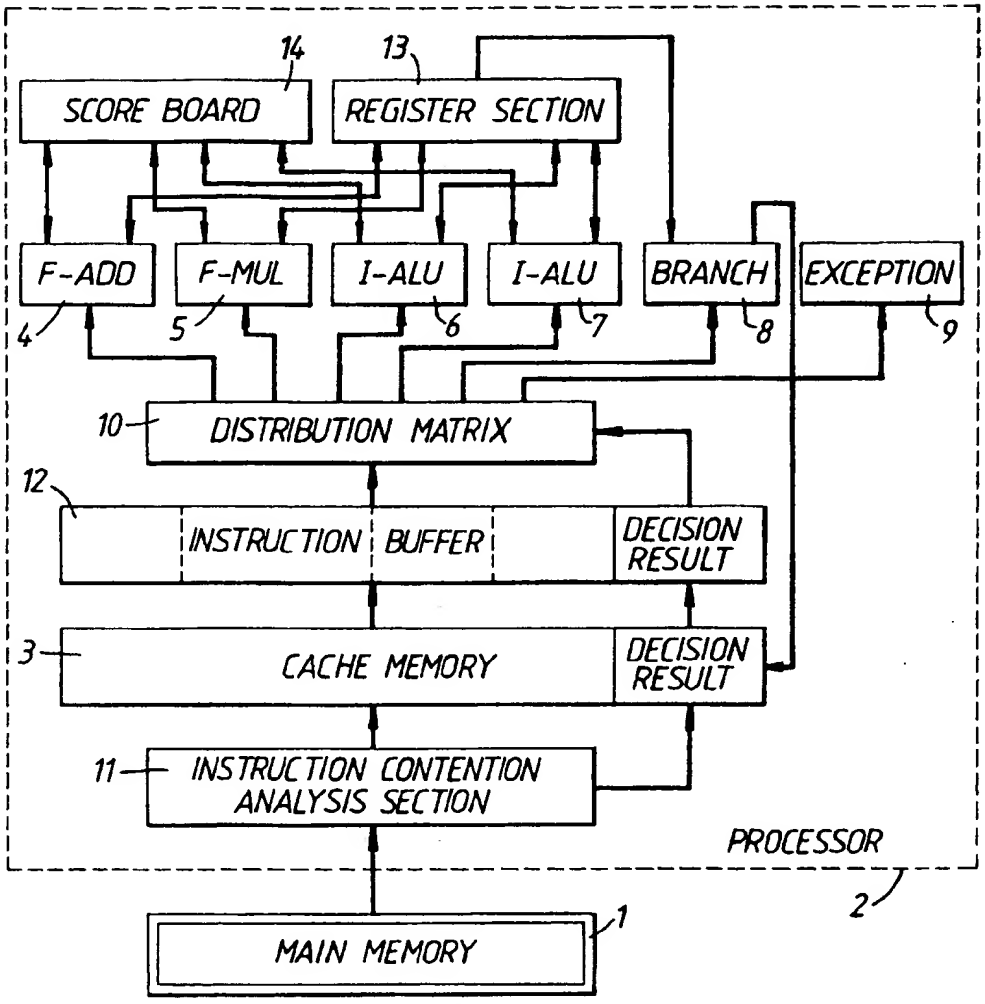
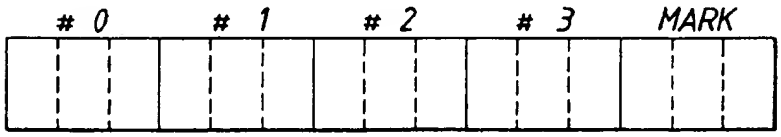


Fig.1.



0 : NOP
1 : IALU 0
2 : IALU 1
3 : BRANCH
4 : EXCEPTION
5 : F-MUL
6 : F-ADD

Fig.2.

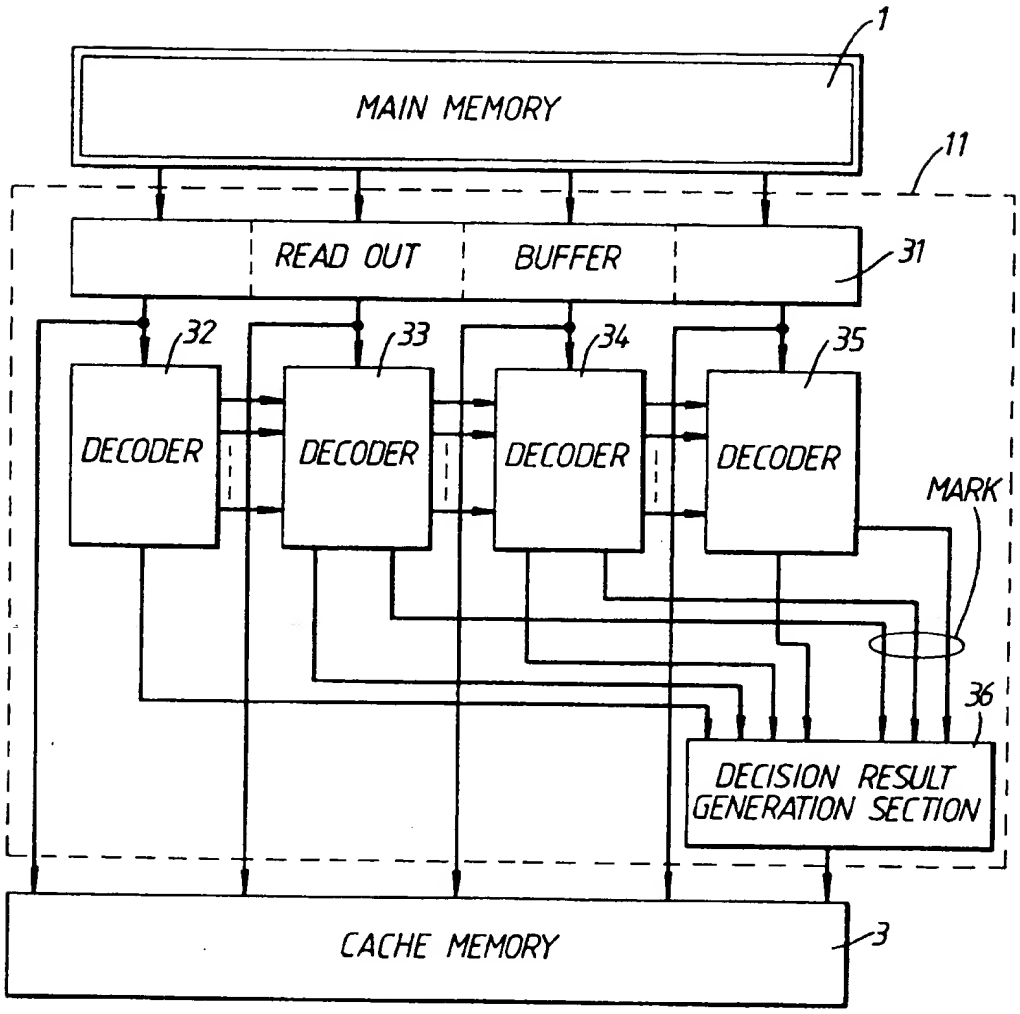

1
MAIN MEMORY
11
READ OUT
BUFFER
31
32
33
34
35
DECODER
DECODER
DECODER
DECODER
MARK
36
DECISION RESULT GENERATION SECTION
CACHE MEMORY
3

Fig.3.

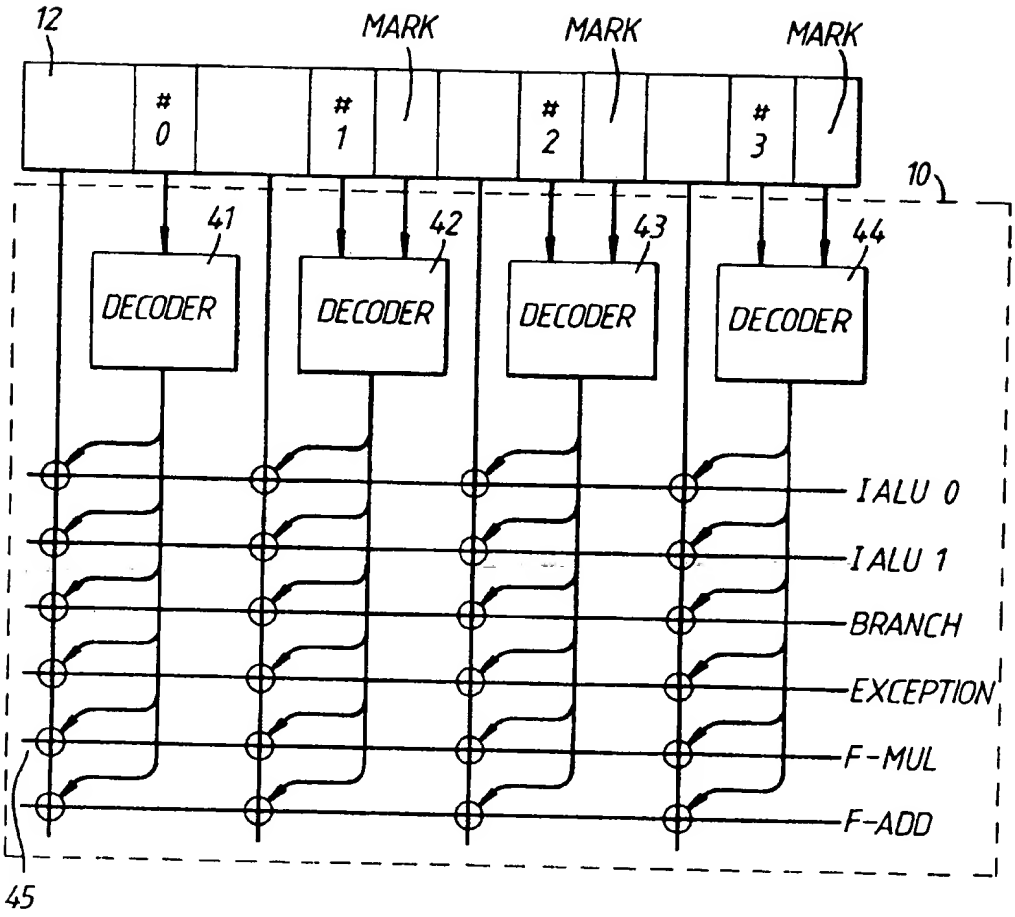

12
MARK
MARK
MARK
0
1
2
3
10
41
42
43
44
DECODER
DECODER
DECODER
DECODER
IALU 0
IALU 1
BRANCH
EXCEPTION
F-MUL
F-ADD
45

Fig. 4.

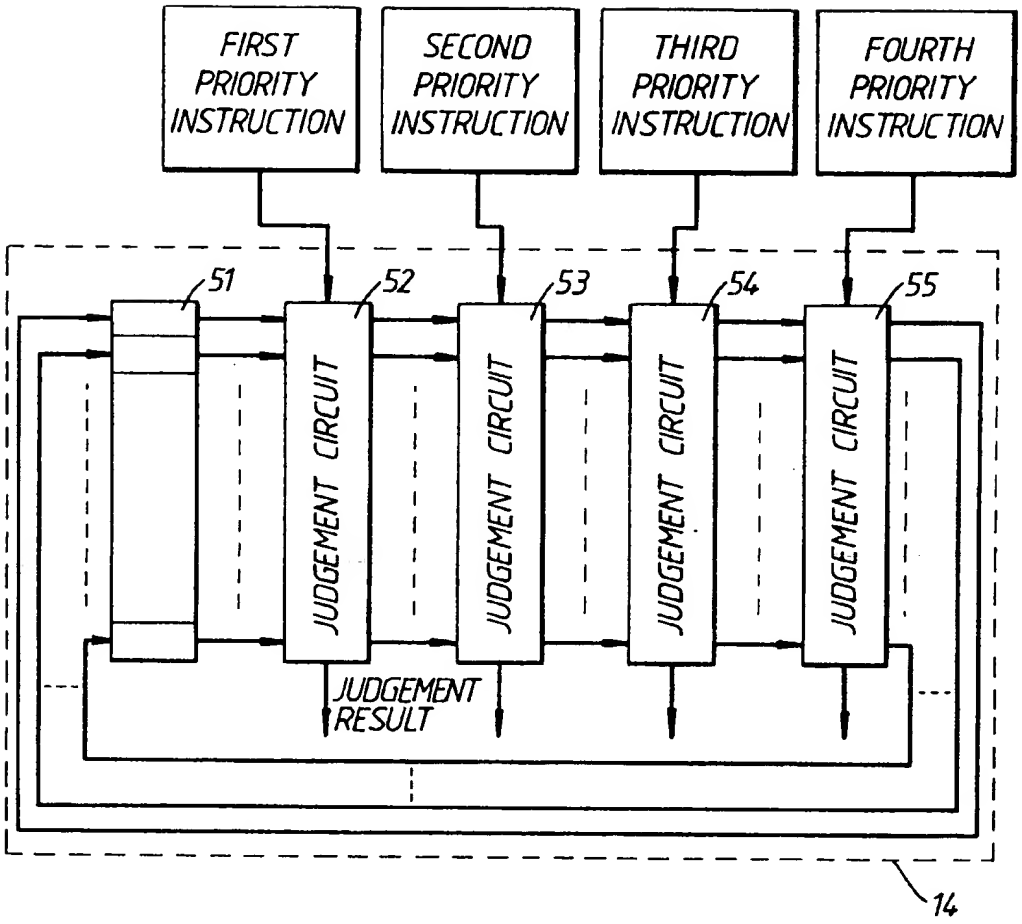

FIRST PRIORITY INSTRUCTION
SECOND PRIORITY INSTRUCTION
THIRD PRIORITY INSTRUCTION
FOURTH PRIORITY INSTRUCTION
51
52
53
54
55
JUDGEMENT CIRCUIT
JUDGEMENT CIRCUIT
JUDGEMENT CIRCUIT
JUDGEMENT CIRCUIT
JUDGEMENT RESULT
14

Fig. 5.

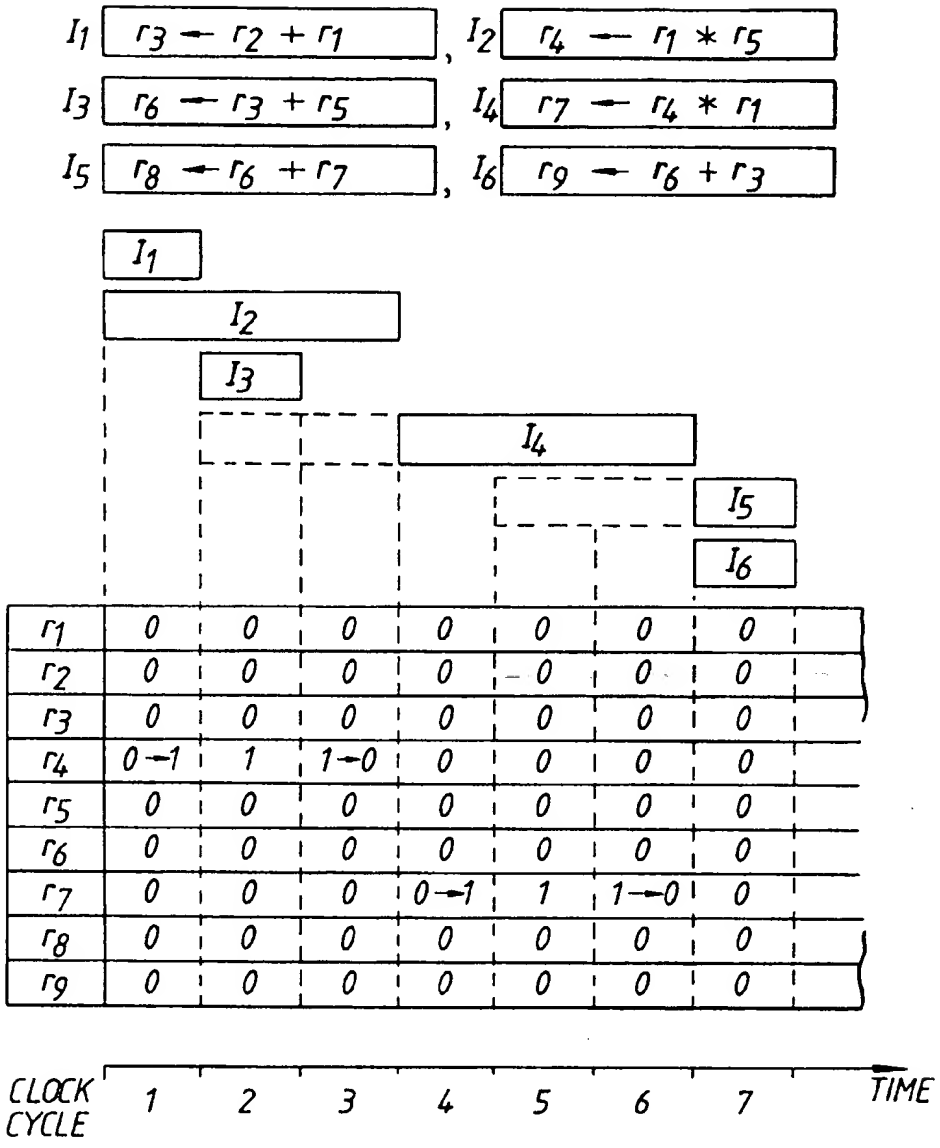

I1 r3 ← r2 + r1 , I2 r4 ← r1 * r5
I3 r6 ← r3 + r5 , I4 r7 ← r4 * r1
I5 r8 ← r6 + r7 , I6 r9 ← r6 + r3
I1
I2
I3
I4
I5
I6
r1 0 0 0 0 0 0 0
r2 0 0 0 0 0 0 0
r3 0 0 0 0 0 0 0
r4 0→1 1 1→0 0 0 0 0
r5 0 0 0 0 0 0 0
r6 0 0 0 0 0 0 0
r7 0 0 0 0→1 1 1→0 0
r8 0 0 0 0 0 0 0
r9 0 0 0 0 0 0 0
CLOCK CYCLE 1 2 3 4 5 6 7 TIME

Fig.6.